PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00 | A2

(11) International Publication Number: WO 95/06285

(43) International Publication Date: 2 March 1995 (02.03.95)

(21) International Application Number: PCT/US94/09723

(22) International Filing Date: 25 August 1994 (25.08.94)

(30) Priority Data:
08/113,417 27 August 1993 (27.08.93) US

(71) Applicant: 3COM CORPORATION [US/US]; 5400 Bayfront Plaza, P.O. Box 58145, Santa Clara, CA 95052-8145 (US).

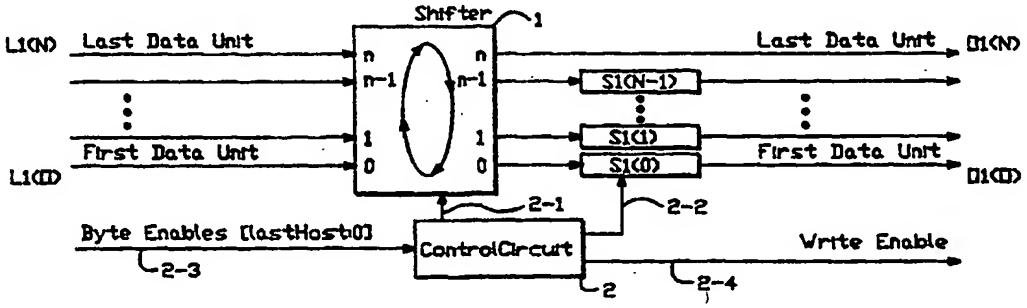
(72) Inventor: PETERSEN, Brian; 731 Vera Cruz Avenue, Los Altos, CA 94022 (US).

(74) Agents: HAYNES, Mark, A. et al.; Haynes & Davis, Suite 310, 2180 Sand Hill Road, Menlo Park, CA 94025-6935 (US).

(81) Designated States: AU, CA, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: READ AND WRITE DATA ALIGNER AND DEVICE UTILIZING SAME



(57) Abstract

A data aligner transfers data from an input having N+1 byte lanes to an output having N+1 byte lanes. The data aligner includes a write data aligner and a read data aligner. The write data aligner includes a write shifter coupled to the N input byte lanes and a stage having N selector/registers S1(i). The N selector/registers each have a queuing register R(i) and bypass multiplexer M(i). The N selector/registers are coupled to the N output byte lanes. The write shifter and N selector/registers S1(i) are coupled to a control circuit. The read data aligner includes a stage having N selector/registers S2(i) and a read shifter. The S2(i) selector/registers are coupled to N+1 byte input lanes with the S2(i) outputs coupled to the N read shifter inputs. The read shifter outputs are then coupled to the N+1 output byte lanes. Finally, a control circuit is coupled to the selector/registers S2(i) and read shifter.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

READ AND WRITE DATA ALIGNER AND DEVICE UTILIZING SAME

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation-in-part of co-pending U.S. patent application entitled DMA DATA PATH ALIGNER AND NETWORK ADAPTOR UTILIZING SAME, Serial No. 07/947,055, filed September 18, 1992, which was owned at the time of invention and is currently owned by the same assignee.

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to the alignment of data segments in a data path. In particular, the present invention relates to the alignment of data segments in transferring data between a host bus and a buffer.

DESCRIPTION OF RELATED ART

In peripheral devices, such as network adapters, high speed sequential access to a first-in-first-out (FIFO) buffer is generally required. In the interest of cost, complexity and minimization, it is generally desirable to implement the FIFO buffer such that all accesses are of a fixed width and alignment. For example, only fixed width 32-bit reads and writes are allowed. However, this places a burden on the driving software because it must assemble the 32-bit words

from non-aligned bytes of data transferred between the FIFO buffer and the host bus.

For example, during a host bus write cycle, only 2 bytes of a 32-bit word in bit positions 8 through 23 of a 32-bit host data bus may be transferred to the buffer. In the next host data bus write cycle, the host bus may transfer 3 more bytes in data bus bit positions 0 through 23.

The driving software would have to copy the first 2 bytes and align them with the first two bytes of the 3 bytes transferred in the second write cycle before writing a full 32 bit aligned word to the buffer. Consequently, the driving software is required to assemble and align bytes of data positioned in various data bus bit positions to form an aligned 32-bit word.

Similarly, a host may require bytes of data in a FIFO buffer to be positioned at certain bit positions on a 32-bit host data bus. For example, a host may require the first 2 bytes of a 32-bit data word in FIFO memory to be positioned at host data bus bit positions 8 through 23. Moreover, the host may desire the last byte of a 32-bit word in FIFO memory to be concatenated transferred with the first byte of the next addressed 32-bit word.

Thus, the driving software is also required to assemble bytes of data from various buffer memory locations and position the bytes on host specified data bus bit positions.

Therefore, it is desirable to provide data alignment logic that allows a fixed width, fixed byte aligned, buffer to be accessed via reads and writes of arbitrary widths and arbitrary byte alignments without requiring additional driving software. In addition, the data alignment logic must not impose performance limitations on the host architecture such as requiring redundant writes or reads, to or from, a single buffer location.

SUMMARY OF THE INVENTION

The present invention provides an apparatus for transferring data from an input data path having N+1 byte lanes to an output data path having N+1 byte lanes. The apparatus comprises a write data aligner and a read data aligner. The write data aligner aligns bytes of data written in a subset of the set of N+1 input byte lanes in order to output bytes of data on the N+1 output byte lanes. The write data aligner outputs bytes on each output byte lane without requiring multiple write cycles per subset of bytes written to the input data path.

The read data aligner outputs bytes of data on a subset of the output byte lanes from bytes of data transferred on each input byte lane. In the subset, each output byte lane transfers a byte of data without requiring multiple read cycles per byte of data read in the subset.

According to one aspect of the invention, the write data aligner includes N+1 input byte lanes L1(i), for i equal to 0 through 3 and N+1 output byte lanes O1(i), for i equal to 0 through 3. Shifting means supplies bytes of data from respective input byte lanes L1(i) to selected shifting means outputs.

The write data aligner also includes a first-stage pipeline. The stage includes N selector/registers S1(i) for staging bytes of data from the shifting means outputs to the N+1 output byte lanes O1(i). The N selector/registers S1(i) in the first stage of the pipeline each have a storage element R(i) for storing a byte of data and a selector M(i). The respective selectors M(i) supply a byte of data from a selected shifting means output or a register R(i) output.

Control means is coupled to the shifting means and the stage for supplying a first and second signal. The control means also includes means for determining the difference between the number of storage elements R(i) storing a byte of data and the number of input byte lanes L1(i) not transferring a byte of data. The shifting means supplies a byte of data to a selected shifting means output in response to the difference.

According to another aspect of the invention, the read data aligner includes an input data path having N+1 byte lanes L2(i) and an output data path having N+1 byte lanes O2(i). The read data aligner also includes a stage of N selector/registers S2(i) having

a storage element R(i) for storing data and selector M(i) for each selector/register S2(i). However, the stage in the read data aligner has inputs coupled to the N input byte lanes L2(i) and outputs supplying bytes of data to a shifting means. The shifting means allows for bytes of data from either the register R(i) outputs or byte lanes L2(i) to be selective supplied to the output byte lanes O2(i).

Control means is coupled to the shifting means and stage supplying a first and second signal. The control means also includes means for determining the number of bytes of data presently outputted from byte lanes O2(i) and a number of bytes of data subsequently outputted from byte lanes O2(i).

According to another aspect of the invention, the write data aligner and read data aligner are coupled to a host bus and a buffer. The N+1 input byte lanes L1(i) are connected to the host bus and the N+1 output byte lanes O1(i) are coupled to a buffer.

Similarly, the read data aligner has N+1 input byte lanes L2(i) coupled to a buffer and N+1 output byte lanes O2(i) coupled to a host bus.

According to another aspect of the invention, a device for transferring data units between a host system and a network is provided. The device includes a bus interface coupled to the host bus and a means for receiving a valid data unit signal from the host system. The device also includes a buffer having a plurality of memory locations for storing each data

unit. The buffer is also coupled to a network controller which transfers the data units to a network. A data path couples the bus interface and the buffer with a data path aligner as described above, provided in the data path.

As can be seen the data aligner of the present invention provides for transferring segments of data of arbitrary width and alignment, such as is necessary for efficient operation of peripheral devices like network adapters, without requiring additional driving software.

Other aspects and advantages of the present invention can be seen upon review of the figures, the detailed description, and the claims which follow.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 is a simplified block diagram of a write data aligner according to the present invention.

Figure 2 is a simplified block diagram of a read data aligner according to the present invention.

Figures 3A-C illustrate the operation of the shifter in Figure 1 according to the present invention.

Figures 4A-C illustrate the operation of the selector/register S1(N-1) in Figure 1 according to the present invention.

Figures 5A-B illustrate the operation of the queuing registers R(i) in selector/register S2(i) in Figure 2 according to the present invention.

Figure 6 illustrates the operation of the shifter in Figure 2 according to the present invention.

Figure 7 is a block diagram of a system with a peripheral device using the data aligner according to the present invention.

Figure 8 is a detailed block diagram of the peripheral device in Figure 7 according to the present invention.

Figure 9 is a block diagram of the write data aligner according to the present invention.

Figure 10 is a logic flow chart of the write data aligner in Figure 9 according to the present invention.

Figure 11 is a timing diagram of the write data aligner in Figure 9 according to the present invention.

Figure 12 is a block diagram of a read data aligner according to the present invention.

Figure 13 is a logic flow chart of the read data aligner in Figure 12 according to the present invention.

Figure 14 is a timing diagram of the read data aligner in Figure 12 according to the present invention.

DETAILED DESCRIPTION

A detailed description of preferred embodiments of the present invention is provided with respect to the Figures.

Figures 1-6 provide a functional overview of the data aligner of the present invention. Figures 1, 3A-C and 4A-C illustrate the operation of the write data aligner. Figures 2, 5A-B and 6 illustrate the operation of the read data aligner. Figure 7 illustrates the application of the invention in a host system. Figures 8-14 provide a detailed description of a peripheral device utilizing the present invention.

I. Functional Overview and Application

A. Write Data Aligner

Figure 1 is a simplified block diagram of a write data aligner. As can be seen in Figure 1, the write data aligner includes a plurality of input segment lanes L1(0) through L1(N). In the preferred embodiment, the N+1 input segment lanes are coupled to a host data bus. Each segment lane L1(0) through L1(N) is used to transfer segments of data from the host data bus to shifter 1. In the preferred embodiment, each lane L1(i), for i equals 0 through i equals 3, includes a byte of data. Accordingly, a 4 byte data word may be transferred on segment lanes L1(0) through L1(3).

The byte of data in one embodiment is 8 bits. The term byte, as used herein, is a familiar terminology to those skilled in the art. The invention may be applied to segments of data of any size, one bit per lane up.

The write data aligner also includes a plurality of output segment lanes 01(0) through 01(N). In the preferred embodiment, the N+1 output segment lanes are coupled to a buffer. Each segment lane 01(0) through 01(N) is used to transfer segments of data from the write data aligner to a buffer. In the preferred embodiment, each lane 01(i), for i equals 0 through i equals 3, includes a byte of data. Accordingly, a 4 byte data word may be transferred on segment lanes 01(0) through 01(N).

In describing the data on segment lanes L1(i), the terms "first data unit" and "last data unit" are used to describe the least significant and most significant data units of a data word being transferred. The terms "least significant" and "most significant" are not used because of two competing processor architectures. Using one processor architecture, a 32-bit data word having four bytes of hexadecimal data 04 03 02 01 stored in a single memory location would be accessed as a series of bytes as follows: 01, 02, 03, 04. The "least significant" byte of data would be accessed first. On the other hand, a second processor architecture would access the same 32-bit data word as the following series of bytes: 04, 03, 02, 01. With regard to the data path aligner, it is not the significance of the data unit or byte that matters, but rather the order in which they are accessed as a sequence of fundamental data units. Therefore, when the term "last data unit" is

used, those using the second processor architecture would translate it to "least significant byte." Those using the first processor architecture would translate "last data unit" to mean "most significant byte."

As mentioned above, there may be circumstances when the host does not write a full valid 32-bit word on the host bus and a full 32-bit word must be assembled and aligned before writing to a buffer. Table 1, seen below, illustrates how bytes of data which are written to input segment lanes L1(0) through L1(3) are outputted on output segment lanes O1(0) through O1(3) according to the present invention. The numbers "XX" under the L1(i) columns represent sequential bytes of data written to segment lanes L1(i) during a host bus write cycle. Each row in Table 1 represents a write cycle having various bytes positioned on various segment lanes. The "- -" under the L1(i) columns represents an invalid or missing data byte written to a L1(i) segment lane. For example, bytes 10 and 11 are written on input segment lanes L1(1) and L1(2), respectively, during the fourth write cycle. Bytes 10 and 11 are aligned with bytes 09 and 12 and outputted on output segment lanes O1(i) to the buffer during the fifth write cycle.

The write data aligner outputs a full 32-bit data word on segment lanes O1(3) through O1(0) from bytes of data placed on various segment lanes L1(i) during various write cycles. As can be seen, the write data

aligner writes to a fixed width buffer from non-aligned and arbitrary width data inputs

Data Written to:				Data Outputted from:			
L1(3)	L1(2)	L1(1)	L1(0)	01(3)	01(2)	01(1)	01(0)
04	03	02	01	04	03	02	01
--	--	--	05				
09	08	07	06	08	07	06	05
--	11	10	--				
12	--	--	--	12	11	10	09
--	13	--	--				
--	--	14	--				
--	--	--	15				
18	17	16	--	16	15	14	13
22	21	20	19	20	19	18	17
--	--	24	23	24	23	22	21

TABLE 1

In the more general case, the data word in a write data aligner may be positioned on data segment lanes 0 through LAST HOST, where 0 is the first host bus segment lane and LAST HOST is the last host bus segment lane. Similarly, LAST STORAGE is the last segment lane on the buffer bus and 0 is the first segment lane on the buffer bus. Using these parameters, the width of the host data bus and buffer data bus would be equal to LAST HOST + 1 and LAST STORAGE + 1, respectively.

The write data aligner in Figure 1 comprises three primary elements: shifter 1, a plurality of first stage selector/registers S1(i), for i equals 0 through i equals N-1 and control circuit 2. Shifter

1 has 0 through N inputs and 0 through N outputs. In the preferred embodiment, each of the plurality of first stage selector/registers S1(i) includes a queuing register R(i) and a bypass multiplexer M(i).

The input data path segment lanes L1(0) through L1(N) are coupled to 0 through N shifter 1 inputs, respectively.

The 0 through N shifter 1 outputs are coupled to the plurality of selector/registers S1(i) for i equals 0 through i equals N-1, respectively. The Nth output of shifter 1 is tied directly to output segment lane O1(N). The selector/registers S1(0) through S1(N-1) are then coupled to output segment lanes O1(0) through O1(N), respectively.

Control circuit 2 is coupled to shifter 1 by line 2-1. Control circuit 2 is also coupled to the plurality of selector/registers S1(0) through S1(N-1) by line 2-2. BYTE ENABLES [LAST HOST:0] signal is input to control circuit 2 on line 2-3. Control circuit 2 then outputs WRITE ENABLE signal on line 2-4.

The following several paragraphs describe the functionality of the elements in the write data aligner.

1. Shifter

The first element in the write data aligner is shifter 1. The purpose of shifter 1 is to place the first valid data unit next to the last data unit in the selector/registers S1(i). For example, in Figure

3A, if two data units 01 and 02 were queued in the queuing registers in selector/register S1(0) and S1(1) and a write contained three valid data units, 03, 04, 05, where the first data unit, 03, was in data segment lane L1(0), then shifter 1 must rotate the data so that the data unit 03 in lane L1(0) will align with selector/register S1(2). This would have the effect of stacking the data units without any gaps.

Figure 3B illustrates L1(0) through L1(N) or LAST HOST being aligned with selector/register S1(i) but all input lanes L1(i) contain valid data. The data units 03 through 06 are rotated to selector/registers S1(2) through S2(N-1). The last two data units 08 and 09 are wrapped around and become aligned with the first two selector/registers S1(0) and S1(1). These data units will be queued for subsequent writing to the data storage element.

Finally, Figure 3C is an example where shifter 1 must shift the data in the opposite direction in order to accomplish the task of assembling gap-free data words. Data units 03 and 04 in data segment lanes L1(N-2) and L1(N-1) must be shifted down to selector/registers S1(2) and S1(3), respectively.

In order to perform the shifting shown in Figures 3A-C, control circuit 2 must make two determinations.

First, the number of input segment lanes L1(i) that are unused must be determined. In other words, the number of segment lanes L1(i) between the first segment lane and the first segment lane transferring

data represents the DATA OFFSET value. In determining the number of segment lanes L1(i) that are unused, there is no need to test the last segment lane. Because, for any valid write cycle, at least one segment lane must carry a data unit.

Second, the number of data units currently held in selector/registers S1(0) through S1(N-1) must be determined. The number of queued data units, or the CURRENT QUEUED value, represent the state value of control circuit 2.

For proper data alignment, the difference between the alignment of the current write and the number of valid data units queued must be eliminated. Hence, the number of positions to rotate the data or the ROTATE value is equal to the difference between the CURRENT QUEUED value and the DATA OFFSET value. For example in Figure 3C, if we assume that LAST STORAGE equals 6, then CURRENT QUEUED value (2) minus DATA OFFSET value (4) equals -2. If the ROTATE value is greater than LAST STORAGE, LAST STORAGE + 1 should be subtracted from the ROTATE value. However, if ROTATE value is less than zero, LAST STORAGE + 1 should be added to ROTATE value. Accordingly, adding seven yields a ROTATE value of five. If the host write data is rotated up and around five positions, the data will be properly aligned.

2. Queuing Registers

As mentioned above, each of the selector/registers S1(i) includes a queuing register or storage element R(i), for i equals 0 through i equals N-1.

The queuing registers R(i) are used to store data between bus write cycles in order to build full data words. Data words are built from the first data unit to the last data unit. The host may write anywhere from 1 through LAST HOST + 1 data words in a single write. The queuing registers R(i) may contain anywhere from 0 through LAST STORAGE data units at any time. If the current write combined with the queued data fails to build a full data word, the write data must be concatenated to the queued data and held until a full data word can be assembled. Likewise, if the combination of the host write data and queued data exceeds LAST STORAGE + 1, then the excess data must be queued for a future write.

In Figure 4A, the combination of current write data (data units 02 and 03) and queued data (data unit 01) is insufficient to build a full data word. Therefore, the two write data units 02 and 03 must be queued in selector/register S1(1) and S1(2), or specifically storage element R(1) and R(2), respectively, until at least one more data unit is written.

In Figure 4B, the data supplied (data units 02, 03 and 04) is exactly enough to build a full data word. Therefore, the data will be immediately written

to the storage element and there is no need to queue the written data in selector/registers S1(1) and S1(2).

Finally, as shown in Figure 4C, the number of data units written (data units 05, 02, 03 and 04) combined with previously queued data (data unit 01) results in an excess of data. The queued data (data unit 01) in selector/register S1(0) along with data units 02, 03, 04 are combined into a full data word and are written into the buffer. The last write data unit 05 must be queued in selector/register S1(0) for eventual combination with further write data.

From Figures 4A-C, three rules of operation for the queuing registers can be derived.

First, if the sum of the number of valid data units in the host write data word and the number of data units currently queued equals LAST STORAGE + 1, then no queuing is required.

Second, if the sum of the number of valid data units in the host write data word and the number of data units currently queued is less than LAST STORAGE + 1, then all of the data units in the host write data word must be queued. The queuing registers that are enabled to receive new data are those just beyond the currently queued data.

Third, if the sum of the number of valid data units in the host write data word and the number of data units currently queued is greater than LAST STORAGE + 1, then those host write data units that are

in excess of those required to build a data word of LAST STORAGE + 1 data units must be queued. In this case, all queuing registers R(i) are enabled.

Finally, control circuit 2 must keep track of how many data units are currently held within the queuing registers R(i).

3. Bypass Multiplexers

As described above, each of the plurality of selector/registers S1(i) include a selector M(i), for i equals 0 through i equals N-1. In the preferred embodiment, the selector is a two input bypass multiplexer.

These bypass multiplexers are used to bypass the respective queuing registers R(i). If the queuing registers are empty and the host is writing a complete data word, then there is no need to perform any shifting and using the queuing registers would introduce alignment errors. Therefore, the queuing registers need to be bypassed.

For a general write case, bypassing queuing registers that contain valid data should not occur.

Finally, a WRITE ENABLE signal for use by the buffer must be generated whenever a full data word has been assembled by the write data aligner. The rules for this function are quite simple: Whenever the sum of the valid host write data units and currently queued data units equal or exceeds the width of the buffer, a write must be enabled.

B. Read Data Aligner

Figure 2 is a simplified block diagram of a read data aligner. Similarly to the write data aligner, the read data aligner has the same structural elements as the write data aligner. However, the purpose of each of the elements are quite different. The purpose of the read data aligner is to align data from a buffer, such that data units appear on the data segment lanes of a host data bus specified by the host. Host reads are allowed to cross buffer word boundaries and may be of arbitrary width and alignment.

The read data aligner includes a plurality of input segment lanes L2(0) through L2(N). In the preferred embodiment, the N+1 segment lanes are coupled to a buffer. Each segment lane L2(0) through L2(N) is used to transfer segments of data from the buffer to selector/registers S2(1) through S2(N). In the preferred embodiment, each lane L2(i), for i equals 0 through i equals 3, includes a byte of data. Accordingly, a 4 byte data word may be transferred on lanes L2(0) through L2(3).

The byte of data in one embodiment is 8 bits. The term byte, as used herein, is a familiar terminology to those skilled in the art. The invention may be applied to segments of data of any size, one bit per lane up.

The read data aligner also includes a plurality of output segment lanes O2(0) through O2(N). In the

preferred embodiment, the N+1 segment lanes are coupled to a host data bus. Each segment lane O2(0) through O2(N) is used to transfer a segment of data from the read data aligner to respective host bus lanes. In the preferred embodiment, each lane O2(i), for i equals 0 through i equals 3, includes a byte of data. Accordingly, a 4 byte data word may be transferred on output segment lanes O2(0) through O2(3).

The read data aligner in Figure 2 comprises three primary elements: a plurality of first stage selector/registers S2(i), for i equals 1 through i equals N, shifter 3 and control circuit 4. In one embodiment, each of the plurality of first stage selector/registers S2(i) includes a queuing register R(i) and a bypass multiplexer M(i). In addition, the read data aligner also includes a shifter 3 having 0 through N inputs and 0 through N outputs.

The input data path segment lanes L2(i), for i equals 1 through i equals N, are coupled to selector/registers S2(i), for i equals 1 through i equals N. The S2(i) selector/registers are coupled to the 1 through N inputs of shifter 3. The input data segment lane L2(0) is coupled directly to shifter 3 input 0. The 0 through N shifter 3 outputs are coupled to the output data segment lanes O2(i), for i equals 0 through i equals N.

Control circuit 4 is coupled to selector/registers S2(i) and shifter 3 by control

lines 4-1 and 4-2, respectively. BYTE ENABLES [LAST HOST:0] signal is input to control circuit 4 on line 4-4. Control circuit 4 outputs READ ENABLE signal on control line 4-2.

1. Queuing Registers

The purpose of the queuing register or storage element R(i) in each selector/register S2(i), for i equals 1 through i equals N, is to preserve the previous read value from the buffer. This allows the read data aligner to build a word from data units from two consecutive buffer words. As soon as the host has read the first data unit of a buffer word, the potential exists for the next read to span two consecutive buffer words. Therefore, the read of the first data unit of the current buffer word must cause the remainder of the current buffer word to be loaded into the queuing registers R(i) while the next data word from the buffer is accessed.

The operation of the queuing registers R(i) is shown in Figures 5A-B. During the first read, data unit 01 is read by the host. Data units 02, 03 and 04 may also be read at the same time by using the bypass multiplexers M(i) discussed below. The read of data unit 01 causes the remainder of the current data word from the buffer to be loaded in queuing registers R(1), R(2) and R(3), respectively.

At the same time, the next storage element data word is accessed and presented to inputs of the read data aligner as seen in Figure 5B.

Now, data units 02, 03 and 04, as well as data units 05, 06, 07 and 08 are available to the host data bus. Therefore, if the next read is a full width read (in this case, 4 data units), then data unit 05 can be included in the word presented to the host bus. Similarly, if the host were to read data units 02 and 03 individually, a full width read would be able to return data units 04, 05, 06 and 07 without having to pause to access a new buffer location or require an additional host read cycle.

Control circuit 4 in Figure 2 requires two pieces of information in order to determine when to queue data and advance to the next buffer data word.

First, control circuit 4 needs to determine how many data units are being requested by the host in a current read cycle or the REQUESTED DATA value. This is done by counting the number of BYTE ENABLE signals that is asserted in BYTE ENABLES [LAST HOST:0] signal generated by the host on line 4-4.

Second, control circuit 4 must keep track of the number of data units that have been read so far. This is done by maintaining a running total of the number of requested data units by the host. The CURRENT READ value is stored in a register that updates the CURRENT READ value with every read cycle by the host. As we are only concerned with the effect of a host

read upon the current buffer data word, once the total CURRENT READ value has exceeded LAST STORAGE, the CURRENT READ value in the register is reduced by LAST STORAGE + 1. Because the queuing of storage element data and the advance of the buffer to its next data word always occur simultaneously, only a single indication needs to be generated. This indication, READ ADVANCE ENABLE signal, is asserted under two conditions: When CURRENT READ value is 0 or if the sum of the CURRENT READ value and REQUESTED DATA value is greater than STORAGE WIDTH + 1.

2. Bypass Multiplexers

If a read data word must be assembled from both queued and current buffer data, then the bypass multiplexers M(i), for i equals 1 through i equals N, are used to bypass the queuing registers R(i), for i equals 1 through i equals N, that contain data that has already been read. The CURRENT READ value that is maintained by control circuit 4 can be used to determine how many of the queuing registers to bypass.

A special case exists when the CURRENT READ value is equal to 0. In this case, all data units that may be read by the host in a current read cycle are on the buffer's data bus; none are in the queuing registers. Therefore, when CURRENT READ value is 0, all of the bypass multiplexers M(i) must be configured to bypass the queuing registers R(i).

3. Shifter

For reads, the purpose of shifter 3 is to align the first unread data unit from the storage element (either queued or not) with the first enabled data lane of the host data bus. Figure 6 illustrates the operation of shifter 3.

The first unread data unit from the buffer is data unit 03. The position of data unit 03 indicates that the CURRENT READ value is equal to 2 because the first two data units 01, 02 (not shown) of the buffer data word have already been read. As illustrated, the host is performing a read on the last three segment lanes of the data bus. For the present example, N equals 3 and the last three output segment lanes 02(3), 02(2) and 02(1) will output data units. This requires a rotation of data units 03, 04 and 05 to output segment lanes O2(1), O2(2) and O2(3), respectively.

II. System Overview

Figure 7 is a schematic diagram of a computer system including a peripheral device having a data aligner according to the present invention. The computer system includes a host system, including a host processor 10, system memory 12, and Direct Memory Access (DMA) controller 13, all communicating through a host system bus 14, such as an EISA bus. The computer bus 14 includes address, control and data lines. Typically, for an EISA bus, there are 32

address lines. Various bus architectures may also include 8, 16, or 32 bi-directional data lines.

DMA controller 13 may be used in the computer system for moving blocks of data from one location to the next, while relieving the host processor 10 of the need to generate a long sequence of addresses to accomplish the move. DMA controller 13 is started by an event, and generates the addresses for moving data from a source location, such as system memory 12, to a destination location, such as peripheral device 15. Typically the data in system memory 12 is a large block of data which begins at a source address, and is moved to a destination beginning at a destination address in peripheral device 15.

Peripheral device 15 with data aligner 24 communicates with host bus 14 through lines 17. In addition, peripheral device 15 communicates with network medium 18, such as an Ethernet network medium, over external interface 16.

Figure 8 provides a block diagram of the peripheral device in Figure 7. Peripheral device 5 is coupled to host bus 15 by address lines 20, control lines 21 and data lines 22 as represented by line 17 in Figure 7.

Peripheral device 15 includes bus interface 23, data aligner 24, memory controller 26, FIFO buffer memory 27 and network controller 25. In various embodiments, there may be numerous other connections

and components not shown having to do with various control and data flow paths.

Bus interface 23 controls the flow of information between host bus 14 and data aligner 24. Host bus 14 is connected to bus interface 23 by address bus 20, control lines 21 and data bus 22.

In the present embodiment, data aligner 24 is coupled between bus interface 23 and memory controller 26. Data transferred to and from the peripheral device is asserted on data bus 42. A BYTE ENABLE signal is asserted on line 41 and a CYCLE STROBE signal is asserted on line 51 to data aligner 24 and memory controller 26. Bus interface 23 is also connected to memory controller 26 by memory address bus 40.

Memory controller 26 controls the flow of data between FIFO buffer memory 27, and data aligner 24 and network controller 25. Memory controller 26 is connected to data aligner 24 by data bus 44 and control line 43. Network controller 25 is coupled to memory controller 26 by data bus 47 and control line 45 and address bus 46. Finally, memory controller 26 is coupled to FIFO buffer memory 27 by data lines 50 and control line 48 and address bus 49.

Network controller 25 provides for the transferring of data on external interface 16 from FIFO buffer memory 27 to a network medium 18, such as an ethernet network.

A. Write Data Aligner Logic

Figure 9 illustrates the implementation of the write data aligner in data aligner 24 of Figure 8. The inputs include byte lanes 42-0, 42-1, 42-2 and 42-3. Each of the input byte lanes may receive 8 bits of write data from bus interface 23. BYTE ENABLE signals are asserted on input lines 41-0, 41-1, 41-2 and 41-3, corresponding to the input byte lanes 42-0, 42-1, 42-2 and 42-3. A BYTE ENABLE signal is asserted on each of the input lines 41 to identify which bytes of data on byte lanes 42 are valid. Input line 41-0 is used for a BYTE ENABLE signal for byte lane 41-0. Input line 41-1 is used for a BYTE ENABLE signal for byte lane 41-1. Input line 41-2 is used for a BYTE ENABLE signal for byte lane 41-2. Finally, input line 41-3 is used for a BYTE ENABLE signal for byte lane 41-3. The two other input lines 41-4 and 51 are used to propagate the HOST WRITE ENABLE signal and the CYCLE STROBE signal, respectively.

The outputs include output byte lanes 44-0, 44-1, 44-2 and 44-3. Each output byte lane may output 8 bits of write data. In addition, MEMORY WRITE ENABLE signal is asserted on output line 43-1.

Barrel shifter 60 has inputs S, 0, 1, 2 and 3 with outputs 0, 1, 2 and 3. Barrel shifter 60 inputs 0, 1, 2 and 3 are connected to input byte lanes 42-0, 42-1, 42-2 and 42-3, respectively. The S input of barrel shifter 60 is connected to control circuit 70 by line 67. Barrel shifter 60 outputs 0, 1, and 2 are

coupled to storage registers 61-0, 61-1 and 61-2, respectively. Barrel shifter 60 output 3 is connected directly to byte lane 44-3.

While D-Q type storage registers 61-0, 61-1 and 61-2 are connected directly to barrel shifter 60 outputs 0, 1 and 2, the barrel shifter outputs 0, 1 and 2 are also connected to the B inputs of bypass multiplexers 62-0, 61-1 and 62-2, respectively. The Q outputs of storage registers 61-0, 61-1 and 61-2 are connected to the A input of bypass multiplexers 62-0, 62-1 and 62-2, respectively. Storage registers 61-0, 61-1 and 61-2 are timed by CYCLE STROBE signal on input line 51 which is connected to the clock inputs of the storage registers. Finally, control circuit 70 enables storage registers 61-0, 61-1 and 61-2 by signals on lines 65-0, 65-1 and 65-2 connected to E inputs of storage registers of 61-0, 61-1 and 61-2, respectively.

Bypass multiplexers 62-0, 62-1 and 62-2 have Y outputs coupled to output byte lanes 44-0, 44-1 and 44-2, respectively. Each Y output may output 8 bits of data. Control circuit 70 is connected to the S inputs of bypass multiplexers 62-0, 62-1, and 62-2 by lines 66-0, 66-1, and 66-2, respectively.

Finally, control circuit 70 is coupled to QUEUE COUNT storage register 69. Data lines 71 and 72 are coupled to the D input and Q output, respectively. Register 69 is clocked by CYCLE STROBE signal on line

51 and enabled by HOST WRITE ENABLE signal on line 41-4 which is connected to the E input.

Figure 10 illustrates the operation of the write data aligner of Figure 9. Although the write data aligner control circuits' operation is depicted as a sequence of steps, in the preferred embodiment, the control circuit is implemented using combinatorial logic or read only memory. After transitioning from block 100, the queue count is reset in block 101. Then, a determination is made whether the HOST WRITE ENABLE signal has been asserted in block 102. If the HOST WRITE ENABLE signal has been asserted, a determination of the offset of the host write data must be determined in block 103; otherwise, the write data aligner will wait for the HOST WRITE ENABLE signal to be asserted. The offset of the host write data in block 103 is ascertained by determining the first valid byte in byte lanes 42-0 through 42-3. For example, if the first valid byte was placed on byte lane 42-1 the data offset value would be equal to 1.

After determining the data offset in block 103, the write data aligner must determine the write count value, or the number of valid bytes on byte lanes 42-0 through 42-3 in block 104.

Block 105 then sets the rotate value to the write count value minus the data offset value.

Block 106 sets the total write data value equal to the sum of the valid bytes on byte lanes 42-0

through 42-3 and the number of bytes queued in storage registers 61-0 through 61-2.

Block 107 then determines whether the total number of bytes or total write data value to be written to the buffer is less than 4. If the value is less than 4, control transitions to block 108; otherwise, the write data aligner transitions to block 109.

If the total write data value is less than 4, storage register 61-0 through 61-2 which are not holding valid write data are enabled in order to allow a write from barrel shifter 60 outputs 0 through 2.

If the total write data value is greater than 4, as determined by block 109, writing to all storage registers 61-0 through 61-2 is enabled in block 110. Otherwise, block 112 sets the next queue count value equal to the total write data value.

If the total write data value is greater than 4 and the write data aligner transitions to block 111, the next queue count value is set to the total write data value minus 4.

Block 113 enables bypass multiplexers 62-0 through 62-2 to bypass storage registers 61-0 through 61-2 which do not contain valid data.

The total write data value is then compared to 4 in block 114. If the value is greater than or equal to 4, MEMORY WRITE ENABLE signal is asserted; otherwise, write data aligner loops back to block 102 for possible further iterations.

Figure 11 illustrates the timing of a typical host write operation. As can be seen from Figure 11, the HOST WRITE DATA signals on byte lanes 42-0 through 42-3 and BYTE ENABLE signals on lines 41-0 through 41-3 are latched on the falling edge of the CYCLE STROBE signal. Also, the MEMORY WRITE DATA signals are outputted to the buffer on byte lanes 44-0 through 44-3 during the low period of the CYCLE STROBE signal.

B. Read Data Aligner Logic

Figure 12 illustrates the implementation of a read data aligner in data aligner 24 in Figure 8. The read data aligner illustrated in Figure 12 aligns data bytes from FIFO buffer memory 27 which are to be positioned on host bus 14. The inputs include byte lanes 44-0, 44-1, 44-2 and 44-3. Each of the input lanes receives a read byte having 8 bits of buffer memory data. In addition, input lines 41-0, 41-1, 41-2 and 41-3, are coupled to control circuit 88. BYTE ENABLE signals are placed on input lines 41-0, 41-1, 41-2 and 41-3 identifying valid bytes on byte lanes 44-0 through 44-3, respectively. Input line 41-0 is used for the BYTE ENABLE signal for byte lane 44-0. Input line 41-1 is used for the BYTE ENABLE signal for byte lane 44-1. Input line 41-2 is used for BYTE ENABLE signal for byte lane 44-2. Finally, input line 41-3 is used for BYTE ENABLE signal for byte lane 44-3. Other inputs to the read data aligner 24 include input line 41-5 which is connected to control circuit

88 and input line 51 which is connected to clock inputs of D-Q type storage registers 81-1, 81-2 and 81-3. The HOST READ ENABLE signal is asserted on input line 41-5 while the CYCLE STROBE signal is asserted on input line 51.

The outputs of the read data aligner include byte lanes 42-0, 42-1, 42-2 and 42-3. Each of the output lanes supplies 8 bits of data to bus interface 23 and eventually to host bus 14. Also control circuit 8 outputs MEMORY READ ENABLE signal on output line 43-2.

Byte lanes 44-1, 44-2 and 44-3 are connected to D inputs of register 81-1, 81-2, 81-3 and B inputs of bypass multiplexers 82-1, 82-2 and 82-3, respectively. The Q outputs of registers 81-1, 81-2 and 81-3 are connected to the A inputs of bypass multiplexers 82-1, 82-2 and 82-3, respectively. Storage registers 81-1, 81-2 and 81-3 are enabled by signals on line 87 connected to control circuit 88 and the E inputs of the storage registers. In addition, line 51 is connected to the clock inputs of registers 81-1, 81-2 and 81-3.

Bypass multiplexers 82-1, 82-2 and 82-3 are connected to control circuit 88 by control lines 84-1, 84-2 and 84-3, respectively. Signals are placed on control lines 84-1, 84-2 and 84-3 in order to select the A input or B input which is outputted from the Y output of bypass multiplexers 82-1, 82-2 and 82-3, respectively.

Barrel shifter 80 has inputs S, 0, 1, 2 and 3. The Y outputs of bypass multiplexers 82-1, 82-2 and 82-3 are connected to barrel shifter inputs 1, 2 and 3, respectively. Byte lane 44-0 is connected directly to input 0 of barrel shifter 80. Control circuit 8 is connected to the S input of barrel shifter 80 by line 85. Barrel shifter 80 also has outputs 0, 1, 2 and 3 connected to output byte lanes 42-0, 42-1, 42-2 and 42-3, respectively.

Queue count register 86 is coupled to control circuit 88 in order to store the queue count value. The queue count value is inputted on line 90 to D input of register 86. Register 86 is clocked by CYCLE STROBE signal on line 51 which is connected to clock input of register 86. Register 86 is enabled by the HOST READ ENABLE signal on line 41-5 which is connected to E input. The queue count is then outputted from Q output on line 91 to control circuit 88.

As with Figure 10, Figure 13 illustrates the operation of the read data aligner of Figure 12. Although the read data aligner control circuits' operation is depicted as a sequence of steps, in the preferred embodiment, the control circuit is implemented using combinatorial logic or read only memory.

After transitioning from block 140, the queue count value is reset in block 141.

The read data aligner then determines whether a HOST READ ENABLE signal is asserted in block 142. If a HOST READ ENABLE signal is asserted, the read data aligner must determine a data offset value of the host read request in block 143. The data offset value is ascertained by determining the first valid output byte lane transferring buffer data in a BYTE ENABLE signal.

Upon determining the data offset value, the read data aligner must determine the number of bytes requested in the host read or read count value from a BYTE ENABLE signal in block 144.

The rotate value for barrel shifter 80 is then determined by subtracting the read count value from the data offset value in block 145.

The total read data value is then set to the sum of the number of valid bytes requested from the host to be read and the queue count value or number of data units stored in storage registers 81-1, 81-2 and 81-3 in block 146.

In block 147, a determination is made whether the total read value equals 0 or the sum of the total read value and requested data value is greater than 4. If either determination is true, the read data aligner transitions to block 148; otherwise, the read data aligner transitions to block 150.

In block 148, writes to storage registers 81-1 through 81-3 are enabled by a signal on line 87.

In addition, the MEMORY READ ENABLE signal is asserted on line 43-2 in order to input another buffer

word on input byte lanes 44-0 through 44-3 in block 149.

Block 150 determines whether the total read data value is greater than or equal to 4. If the total read data value is greater than or equal to 4 the read data aligner transitions to block 151; otherwise, the read data aligner transitions to block 152.

In block 151, the next queue count value is set to the total read data value minus 4. If the total read data value is not greater than or equal to 4 the next queue count value is set to the total read data value in block 152.

Finally, block 153 enables bypass multiplexers 82-1 through 82-3 to bypass storage registers 81-1 through 81-3 which have already been read. The read data aligner then loops back to block 142.

Figure 14 illustrates the timing of the read data aligner in Figure 12. As can be seen from Figure 14, HOST READ ENABLE signal and BYTE ENABLE signals initiates the operation of the read data aligner. After the BYTE ENABLE signals and HOST READ ENABLE signal are generated from the host, the MEMORY READ ENABLE signal is asserted on line 43-2 and MEMORY READ DATA signals are outputted on byte lanes 44-0 through 44-3. Finally the HOST READ DATA signals are asserted on byte lanes 42-0 through 42-3.

III. Conclusion

In conclusion, the present invention provides a data aligner having a write data aligner and a read data aligner. The write data aligner allows for writing a fixed width word to a buffer from bytes of data on a host data bus. The write data aligner assembles bytes of data positioned on various byte lanes on the host data bus and aligns a fixed width word which is written to the buffer. The read data aligner allows for aligning read data from a buffer such that the bytes of buffer data appear on host data bus byte lanes specified by the host. Host reads may be allowed to cross buffer word boundaries and may be of any arbitrary width and alignment.

The present invention further provides for a peripheral device, such as a network adapter, using a data aligner in both the read and write paths in a host system. The read and write data aligners allow for the elimination of driver software which would build and align bytes of data, while not imposing any performance limitations on the host system architecture. The host system is not required to perform redundant accesses (reads or writes) either across the host system bus or to the buffer.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to

practitioners skilled in this art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

CLAIMS

1. An apparatus for transferring data from an input data path having N+1 segment lanes L1(i), for i equal to 0 through N, each segment lane for transferring a segment of data, to an output data path having N+1 segment lanes O1(i), for i equal to 0 through N, each segment lane for transferring a segment of data, comprising:

control means for supplying a first signal and a second signal;

shifting means, coupled to the control means, having N+1 inputs coupled to the N+1 segment lanes L1(i) and having N+1 outputs, for supplying a segment of data from a selected segment lane L1(i) to a selected shifting means output in response to the first control signal; and

a stage, coupled to the control means, including N elements S1(i) for i equal to 0 to N-1, the N elements each having an input coupled to respective N+1 shifting means outputs and the N elements each having an output coupled to respective N segment lanes O1(i),for i equal to 0 through N-1, having respective outputs supplying segments of data selected from the respective shifting means outputs to respective segment lanes O1(i) in response to the second control signal.

2. The apparatus of Claim 1, wherein the N elements S1(i) include:

N storage elements R(i) for i equal to 0 to N-1, storage element R(i) having an input connected to a respective shifting means output and having an output, for storing a segment of data for supplying to the respective output segment lane O1(i); and

N selectors M(i) for i equals to 0 to N-1, selector M(i) having a first input connected to a respective shifting means output and a second input connected to the respective storage element R(i) output, and having an output supplying a segment of data to a respective output segment lane O1(i) in response to the second signal.

3. The apparatus of Claim 2, wherein the means for supplying a first signal includes means for determining the difference between the number of storage elements R(i) storing a segment of data and the number of segment lanes L1(i) not transferring a segment of data.

4. The apparatus of Claim 3, wherein the shifting means supplies a segment of data to a selected shifting means output in response to the difference.

5. The apparatus of Claim 2, wherein means for supplying the second signal includes means for determining the sum between the number of segment lanes L1(i) transferring a segment of data and the number of storage elements R(i) storing a segment of data.

6. The apparatus of Claim 5, wherein the means for supplying the second signal is coupled to the selectors M(i) enabling the segments of data outputted from the N shifting means outputs to be selectively supplied to the N segment lanes O1(i) in response to the sum.

7. The apparatus of Claim 5, wherein the means for supplying a second signal is coupled to the storage elements R(i) enabling the segments of data outputted from the N shifting means outputs to be selectively stored in the respective storage elements R(i) responsive to the sum.

8. The apparatus of Claim 1, wherein each segment lane consists of a byte of data, and N equals 3.

9. The apparatus of Claim 1, wherein a shifting means output is connected to segment lanes O1(N).

10. An apparatus for transferring data from an input data path having N+1 segment lanes L2(i), for i equal to 0 through N, each segment lane for transferring a segment of data, to an output data path having N+1 segment lanes O2(i), for i equal to 0 through N, each segment lane for transferring a segment of data, comprising:

control means for supplying a first signal and a second signal;

a stage, coupled to the control means, including N elements S2(i) for i equal to 1 to N, the N elements each having an input coupled to respective N segment lanes L2(i) and the N elements each having an output for i equal to 1 through N, having respective outputs supplying segments of data selected from the respective segment lanes L2(i) in response to a first signal; and

shifting means, coupled to the control means, having N+1 inputs coupled to the respective N element outputs and having N+1 outputs coupled to respective segment lanes O2(i), for supplying a segment of data from an element S2(i) output to a selected output segment lane in response to the second signal.

11. The apparatus of Claim 10, wherein the N elements S2(i) include:

N storage elements R(i) for i equal to 0 to N-1, storage element R(i) having an input connected to segment lane L2(i) and having an output, for storing a segment of data for supplying the respective shifting means input; and

N selectors M(i) for i equal to 1 to N, selector M(i) having a first input connected to a segment lane L2(i) and a second input connected to the respective storage element R(i) output, and an output supplying a segment of data to a respective shifting means input in response to the first signal.

12. The apparatus of Claim 11, wherein the means for supplying a first control signal includes means for determining the number of segment lanes O2(i) presently outputting a segment of data and the number of segment lanes O2(i) subsequently outputting a segment of data.

13. The apparatus of Claim 12, wherein means for supplying a first signal includes means for determining the sum between the number of segment lanes O2(i) presently outputting a data segment and the number of segment lanes O2(i) subsequently outputting a data segment.

14. The apparatus of Claim 13, wherein the means for supplying a first signal is coupled to storage element R(i) allowing for a segment of data from the segment lane L2(i) to be selectively stored in storage elements R(i) responsive to the sum.

15. The apparatus of Claim 12, wherein the means for supplying a second signal is coupled to the shifter means allowing for a segment of data from the R(i) output or segment lane L2(i) to be selectively supplied to segment lane O2(i).

16. The apparatus of Claim 10, wherein each segment lane consists of a byte of data, and N equals 3.

17. The apparatus of Claim 10, wherein a shifting means input is connected to segment lane L1(0).

18. An apparatus for transferring data from a host bus, having a bus write cycle, to a buffer, comprising:

an input data path, for connecting to the host bus, having N+1 segment lanes L1(i), for i equal to 0 through N, a subset of the set of N+1 segment lanes transferring data, wherein each segment lane in the subset transfers a segment of data;

an output data path for connecting to the buffer, having N+1 segment lanes O1(i) for transferring a segment of data;

means for receiving a segment lane enable signal; and

means, connected between the input and output data paths for controlling alignment of segments of data in the subset in response to the enable signal so that N+1 segment lanes O(i) transfers aligned segments of data to the buffer without requiring multiple bus write cycles per transfer of segments of data in the subset.

19. The apparatus of Claim 18, wherein the means for controlling alignment includes:

control means for supplying a first signal and a second signal responsive to the enable signal;

shifting means, coupled to the control means, having N+1 inputs coupled to the N+1 segment lanes L1(i) and having N+1 outputs, for supplying a segment of data from a selected segment lane L1(i) to a selected shifting means output in response to the first control signal; and

a stage, coupled to the control means, including N elements S1(i) for i equal to 0 to N-1, the N elements each having an input coupled to respective N+1 shifting means outputs and each having an output coupled to respective N segment lanes O1(i), for i equal to 0 through N-1, having respective outputs

supplying segments of data selected from the respective shifting means outputs to respective segment lanes O(i) in response to a second control signal.

20. The apparatus of Claim 19, wherein the N elements S1(i) include:

N storage elements R(i) for i equal to 0 to N-1, storage element R(i) having an input connected to a respective shifting means output and having an output, for storing a segment of data for supplying to the respective output segment lane O1(i); and

N selectors M(i) for i equals to 0 to N-1, selector M(i) having a first input connected to a respective shifting means output and a second input connected to the respective storage element R(i) output, and having an output supplying a segment of data to a respective output segment lane O1(i) in response to the second signal.

21. The apparatus of Claim 20, wherein the means for supplying a first signal includes means for determining the difference between the number of storage elements R(i) storing a segment of data and the number of segment lanes L1(i) not transferring a segment of data.

22. The apparatus of Claim 21, wherein the shifting means supplies a segment of data to a selected shifting means output in response to the difference.

23. The apparatus of Claim 19, wherein means for supplying the second signal includes means for determining the sum between the number of segment lanes L1(i) transferring a segment of data and the number of storage elements R(i) storing a segment of data.

24. The apparatus of Claim 23, wherein the means for supplying the second signal is coupled to the selectors M(i) enabling the segments of data outputted from the N shifting means outputs to be selectively supplied to the N segment lanes O1(i) in response to the sum.

25. The apparatus of Claim 23, wherein the means for supplying a second signal is coupled to the storage elements R(i) enabling the segments of data outputted from the N shifting means outputs to be selectively stored in the respective storage elements R(i) responsive to the sum.

26. The apparatus of Claim 18, wherein each segment lane consists of a byte of data, and N equals 3.

27. An apparatus for transferring data from a buffer to a host bus having a bus read cycle, comprising:

an input data path, for connecting to the buffer, having N+1 segment lanes L2(i), for i equal to 0 through N, each segment lane transferring a segment of data;

an output data path, for connecting to the host bus, having N segment lanes O2(i), for i equal to 0 through N, a subset of the set of N segment lanes transferring data, wherein each segment lane in the subset transfers a segment of data;

means for receiving a segment lane enable signal; and

means, connected between the input and output data paths, for controlling the alignment of segments of data in the subset of segments of data transferred on the N+1 segment lanes L2(i) in response to the enable signal so that the subset of the set of N segment lanes aligns data segments to the host bus without requiring multiple bus read cycles per transfer of data in the subset.

28. The apparatus of Claim 27, wherein the means for controlling alignment includes:

control means for supplying a first signal and a second signal responsive to the enable signal;

a stage, coupled to the control means, including N elements S2(i) for i equal to 1 to N, the N elements

each having an input coupled to respective N segment lanes L2(i) and N elements each having an output for i equal to 1 through N, having respective outputs supplying segments of data selected from the respective segment lanes L2(i) in response to a first signal; and

shifting means, coupled to the control means, having N+1 inputs coupled to the respective N element outputs and having N+1 outputs coupled to respective segment lanes O2(i), for supplying a segment of data from a respective element S2(i) output to a selected output segment lane output in response to the second signal.

29. The apparatus of Claim 28, wherein the N elements S2(i) include:

N storage elements R(i) for i equal to 0 to N-1, storage element R(i) having an input connected to segment lane L2(i) and having an output, for storing a segment of data for supplying the respective shifting means input; and

N selectors M(i) for i equal to 1 to N, selector M(i) having a first input connected to a segment lane L2(i) and a second input connected to the respective storage element R(i) output, and an output supplying a segment of data to a respective shifting means input in response to the first signal.

30. The apparatus of Claim 29, wherein the means for supplying a first control signal includes means for determining the number segment lanes O2(i) presently outputting a data segment and the number of segment lanes O2(i) subsequently outputting a data segment.

31. The apparatus of Claim 30, wherein means for supplying a first signal includes means for determining the sum between the number of segment lanes O2(i) presently outputting a segment of data and the number of segment lanes O2(i) subsequently outputting a segment of data.

32. The apparatus of Claim 31, wherein the means for supplying a first signal is coupled to a storage element R(i) allowing for a segment of data from a segment lane L2(i) to be selectively stored in the storage element R(i) responsive to the sum.

33. The apparatus of Claim 29, wherein the means for supplying a second signal is coupled to the shifter means allowing for a segment of data from a R(i) output or a segment lane L2(i) to be selectively supplied to a segment lane O2(i).

34. The apparatus of Claim 27, wherein each segment lane consists of a byte of data, and N equals 3.

36. A device for transferring data units from a network to a host system having a host bus having a plurality of segment data lanes, comprising:

a bus interface, coupled to the host bus, for transferring a data unit on each segment data lane during a bus read cycle;

means for receiving a valid data unit signal from the host system;

a buffer, having a plurality of memory locations, for storing each data unit;

a data path between the bus interface and the buffer;

a network controller, coupled to the buffer and the network, for transferring data units between the network and the buffer; and

means, in the data path, for controlling alignment of each data unit in the data path in response to the valid data unit signal so that the data units are transferred on a subset of the plurality of segment data lanes, each segment lane in the subset transferring a data unit, without requiring multiple bus read cycles per transferred data units in the subset.

35. A device for transferring data units from a host system, having a host bus having a plurality of segment data lanes, to a network, comprising:

a bus interface, coupled to the host bus, for transferring a data unit on each segment data lane in a subset of the plurality of segment data lanes during a bus write cycle;

means for receiving a valid data unit signal from the host system;

a buffer, having a plurality of memory locations, for storing each data unit;

a data path between the bus interface and the buffer;

a network controller, coupled to the buffer and the network, for transferring a data unit between the buffer and the network; and

means, in the data path, for controlling alignment of each data unit in the data path in response to the valid data unit signal so that the data units transferred in the subset are sequentially stored in the plurality of buffer memory locations without requiring multiple bus write cycles per transferring of data units in the subset.

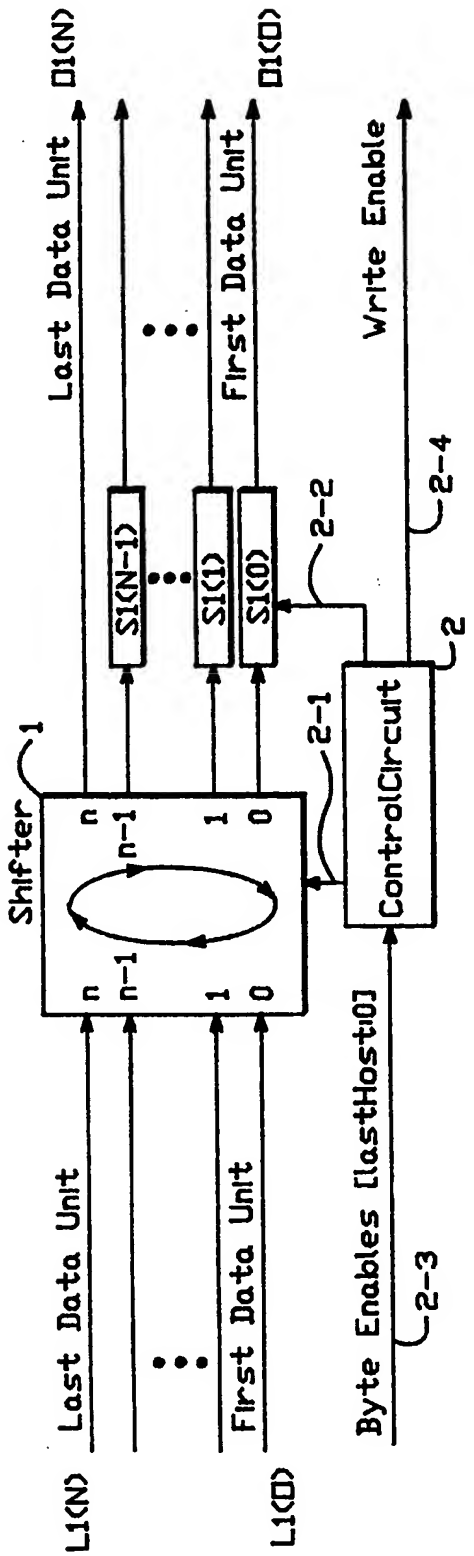

L1(N)
Last Data Unit
L1(0)
First Data Unit
Byte Enables [lastHost:0]
2-3
Write Enable
2-4
Shifter
1
n
n-1
1
0
S1(N-1)
S1(1)
S1(0)
2-2
2-1
ControlCircuit
2
D1(N)
Last Data Unit
D1(0)
First Data Unit

FIG. 1

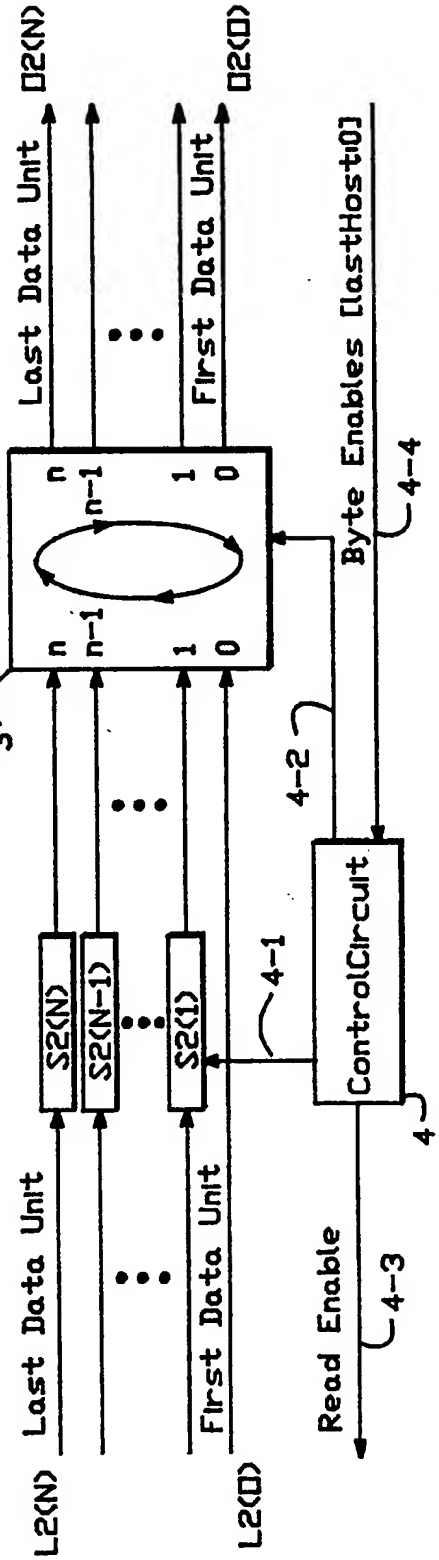

L2(N)
Last Data Unit
L2(0)
First Data Unit
Read Enable
4-3
Byte Enables [lastHost:0]
4-4
Shifter
3
n
n-1
1
0
S2(N)
S2(N-1)
S2(1)
4-1
4-2
ControlCircuit
4
D2(N)
Last Data Unit
D2(0)
First Data Unit

FIG. 2

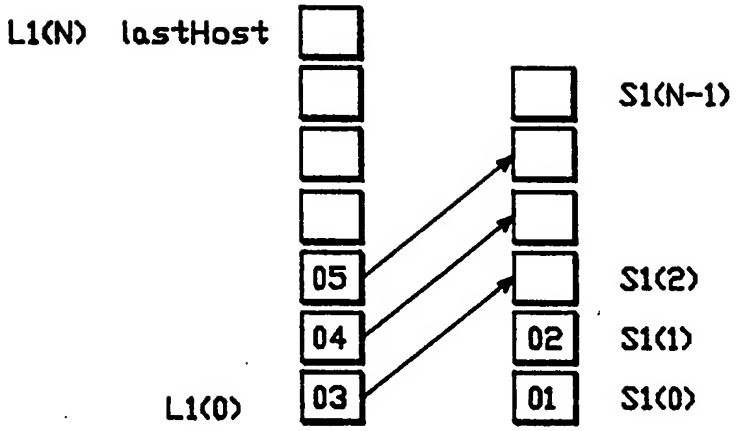


FIG.3A

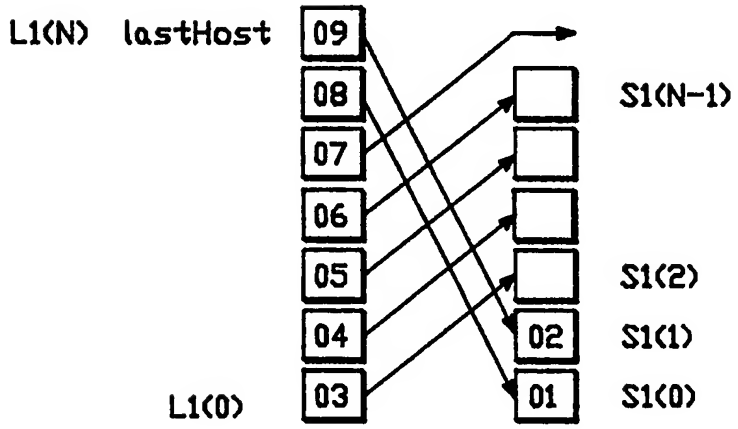


FIG.3B

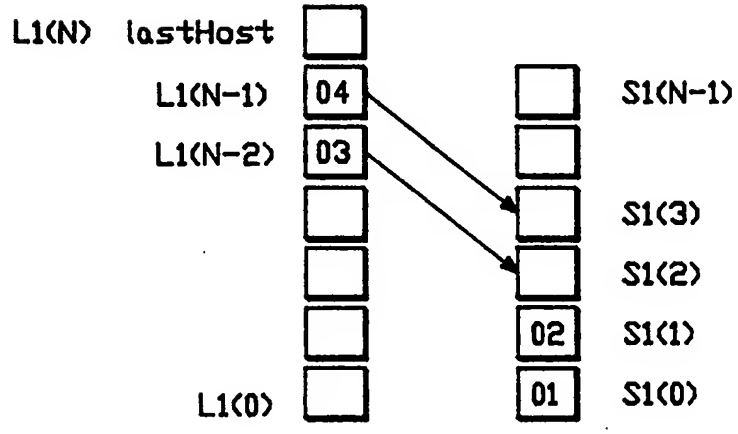


FIG.3C

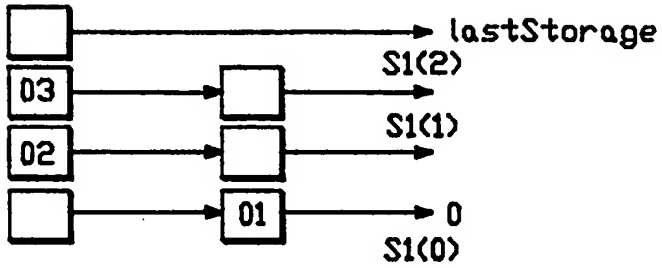


FIG.4A

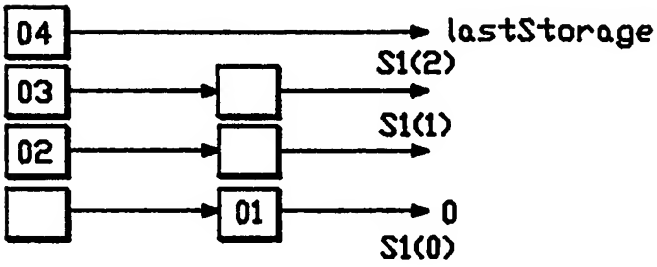


FIG.4B

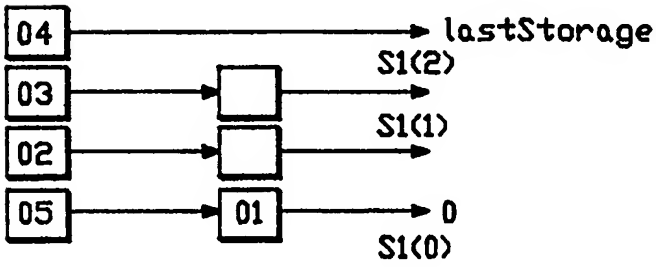


FIG.4C

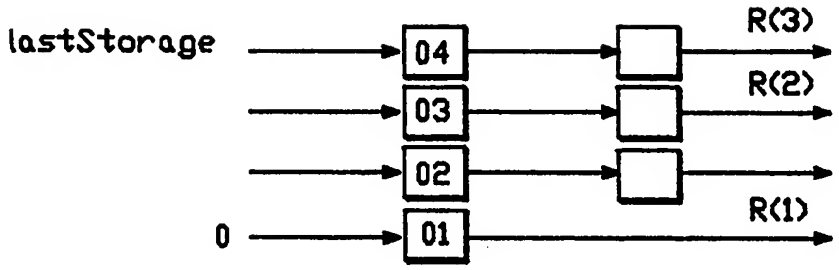


FIG.5A

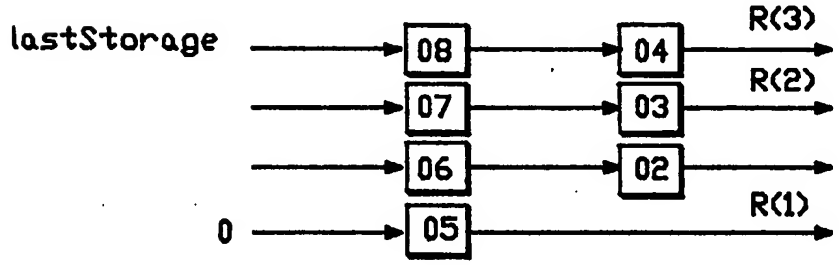


FIG.5B

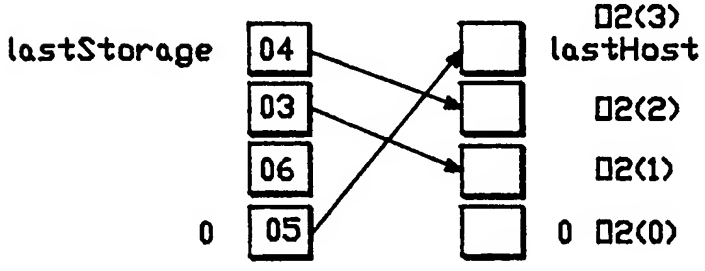


FIG.6

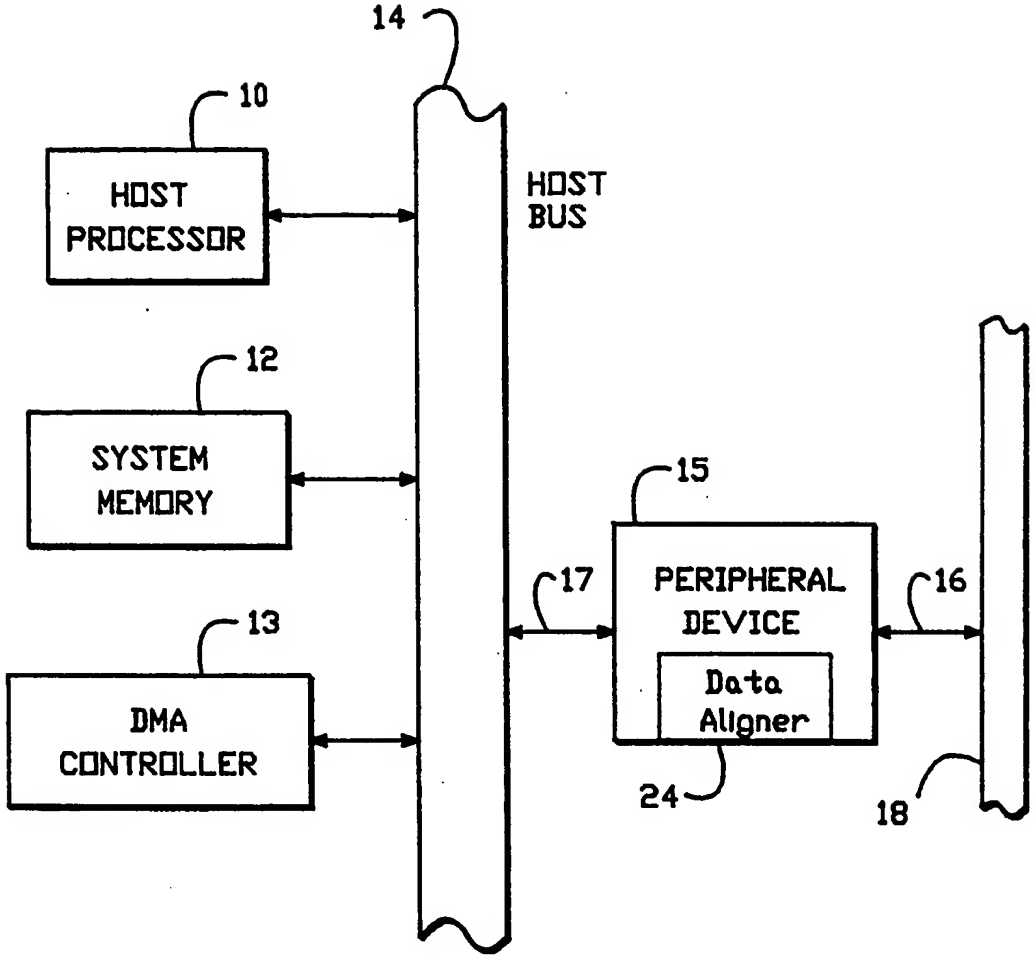

14
10
HOST PROCESSOR
HOST BUS
12
SYSTEM MEMORY
15
17
PERIPHERAL DEVICE
16
13
DMA CONTROLLER
Data Aligner
24
18

FIG.7

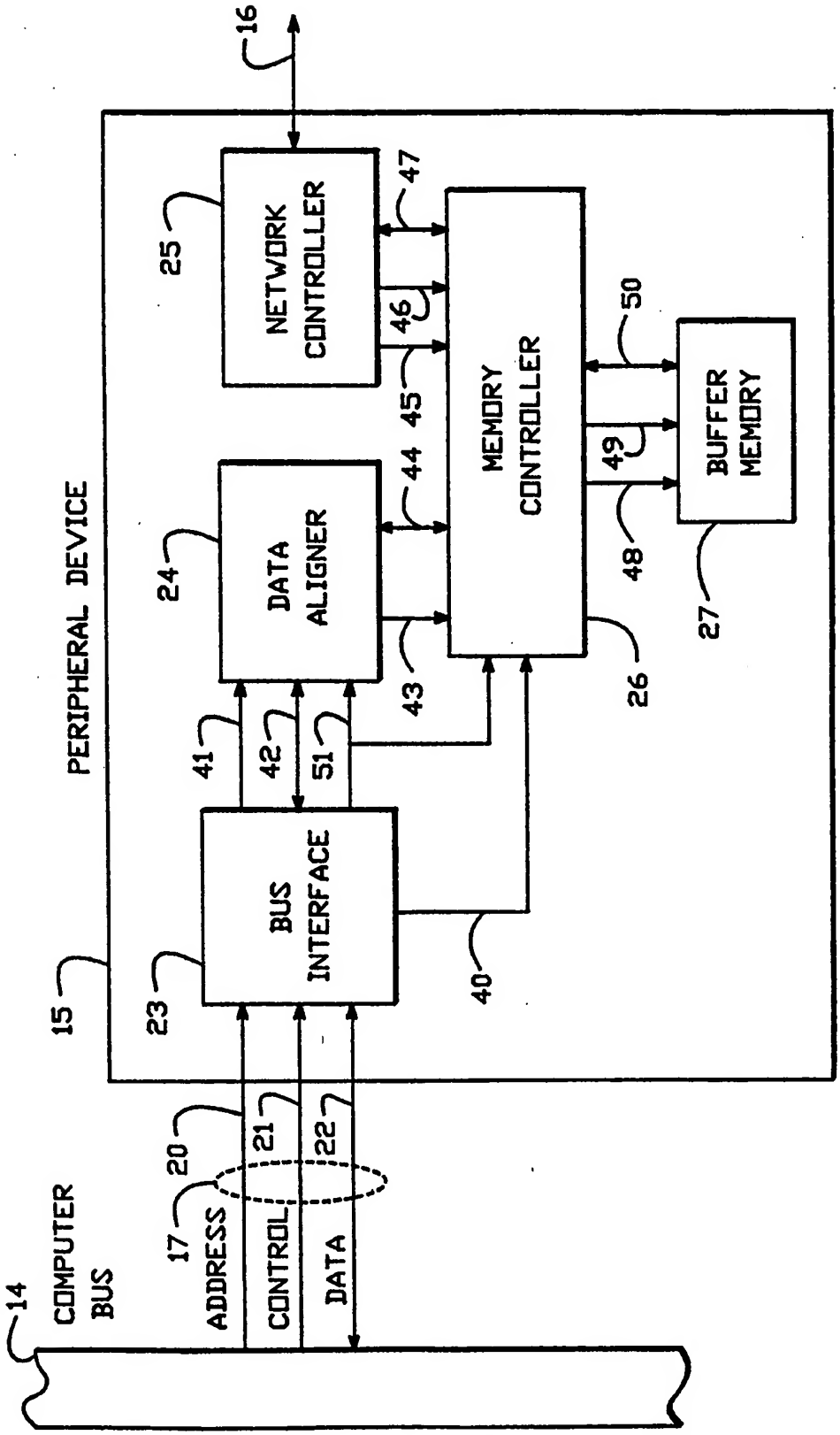

16
25
NETWORK CONTROLLER
47
46
45
44
MEMORY CONTROLLER
50
49
48
BUFFER MEMORY
27
26
24
DATA ALIGNER
43
41
42
51
PERIPHERAL DEVICE
15
23
BUS INTERFACE
40
20
21
22
17
ADDRESS
CONTROL
DATA
14
COMPUTER BUS

FIG.8

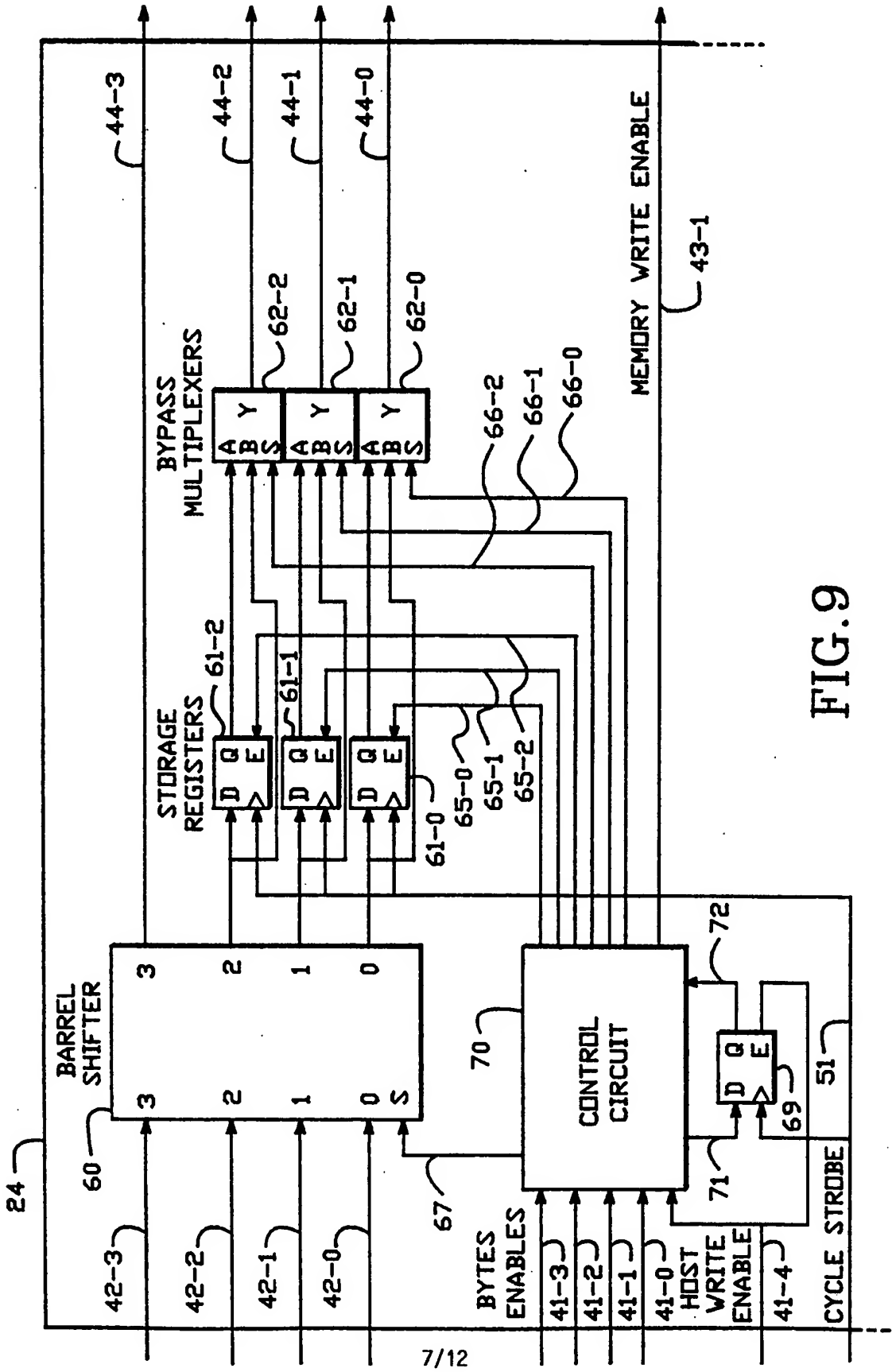

24
60
BARREL SHIFTER
3
2
1
0
S
42-3
42-2
42-1
42-0
44-3
44-2
44-1
44-0
BYPASS MULTIPLEXERS
A B S Y
62-2
62-1
62-0
STORAGE REGISTERS
D Q E
61-2
61-1
61-0
65-0
65-1
65-2
66-2
66-1
66-0
MEMORY WRITE ENABLE
43-1
67
70
CONTROL CIRCUIT
72
71
69
51
BYTES ENABLES
41-3
41-2
41-1
41-0
HOST WRITE ENABLE
41-4
CYCLE STROBE

FIG.9

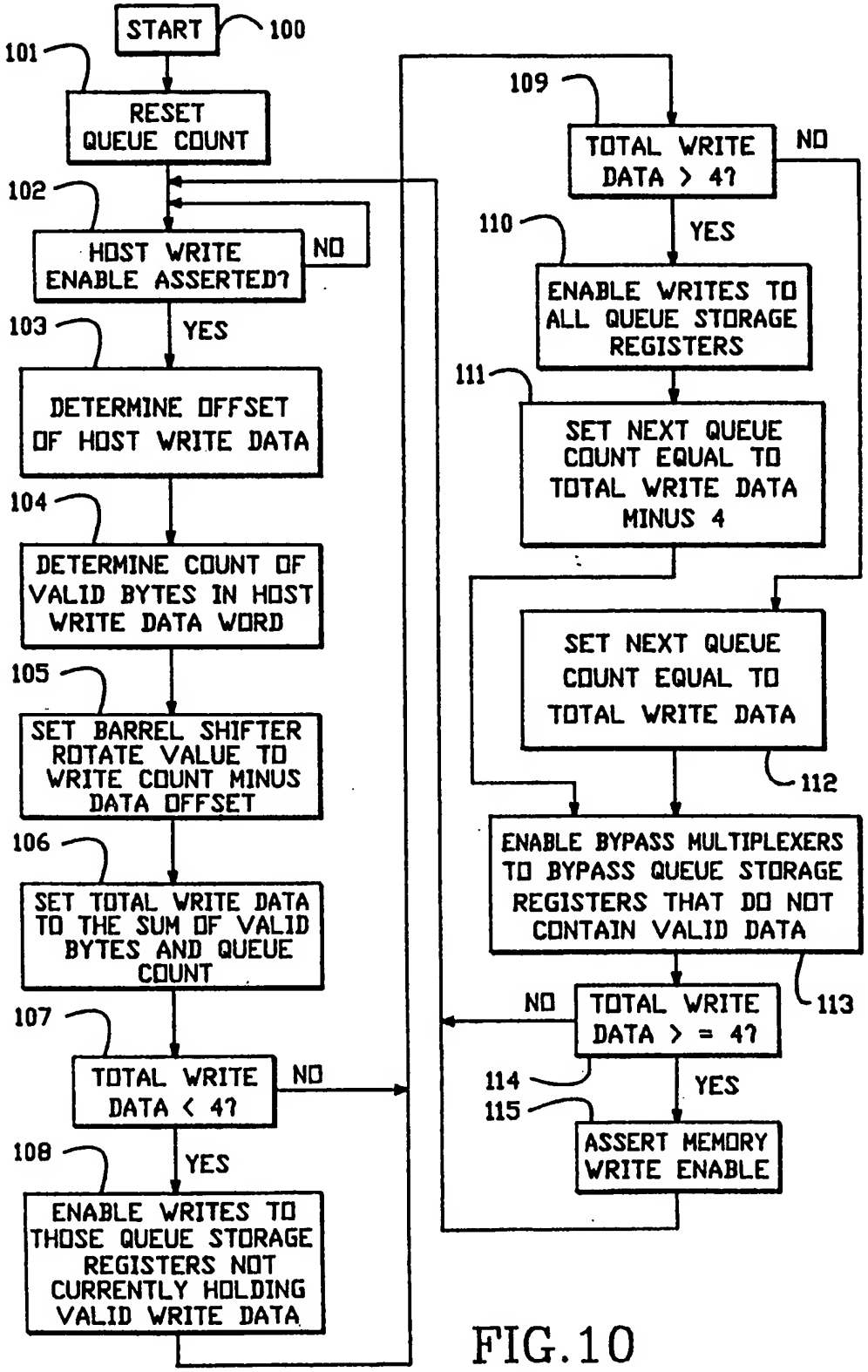

START
100
101
RESET QUEUE COUNT
102
HOST WRITE ENABLE ASSERTED?
NO
YES
103
DETERMINE OFFSET OF HOST WRITE DATA
104
DETERMINE COUNT OF VALID BYTES IN HOST WRITE DATA WORD
105
SET BARREL SHIFTER ROTATE VALUE TO WRITE COUNT MINUS DATA OFFSET
106
SET TOTAL WRITE DATA TO THE SUM OF VALID BYTES AND QUEUE COUNT
107
TOTAL WRITE DATA < 4?
NO
YES
108
ENABLE WRITES TO THOSE QUEUE STORAGE REGISTERS NOT CURRENTLY HOLDING VALID WRITE DATA
109
TOTAL WRITE DATA > 4?
NO
YES
110
ENABLE WRITES TO ALL QUEUE STORAGE REGISTERS
111
SET NEXT QUEUE COUNT EQUAL TO TOTAL WRITE DATA MINUS 4
SET NEXT QUEUE COUNT EQUAL TO TOTAL WRITE DATA
112
ENABLE BYPASS MULTIPLEXERS TO BYPASS QUEUE STORAGE REGISTERS THAT DO NOT CONTAIN VALID DATA
113
TOTAL WRITE DATA > = 4?
NO
YES
114
115
ASSERT MEMORY WRITE ENABLE

FIG.10

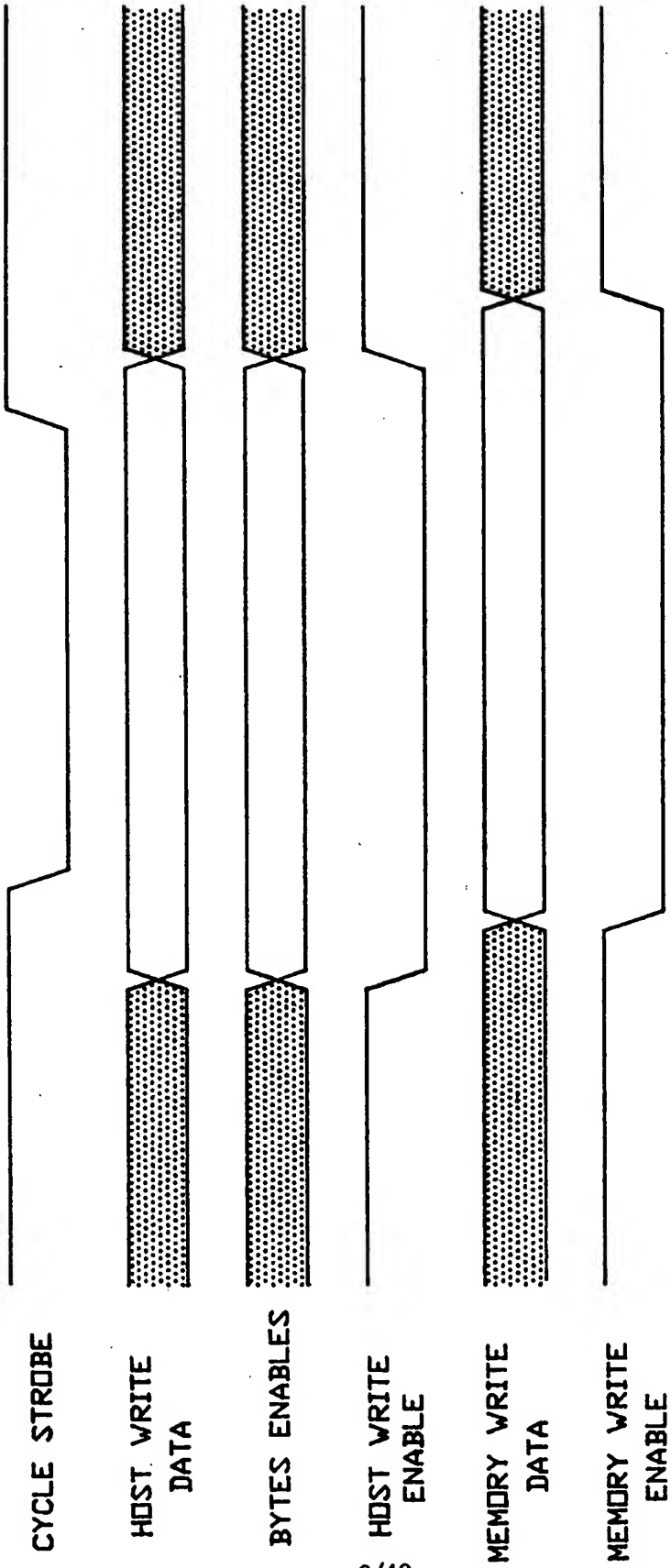

CYCLE STROBE
HOST WRITE DATA
BYTES ENABLES
HOST WRITE ENABLE
MEMORY WRITE DATA
MEMORY WRITE ENABLE

FIG.11

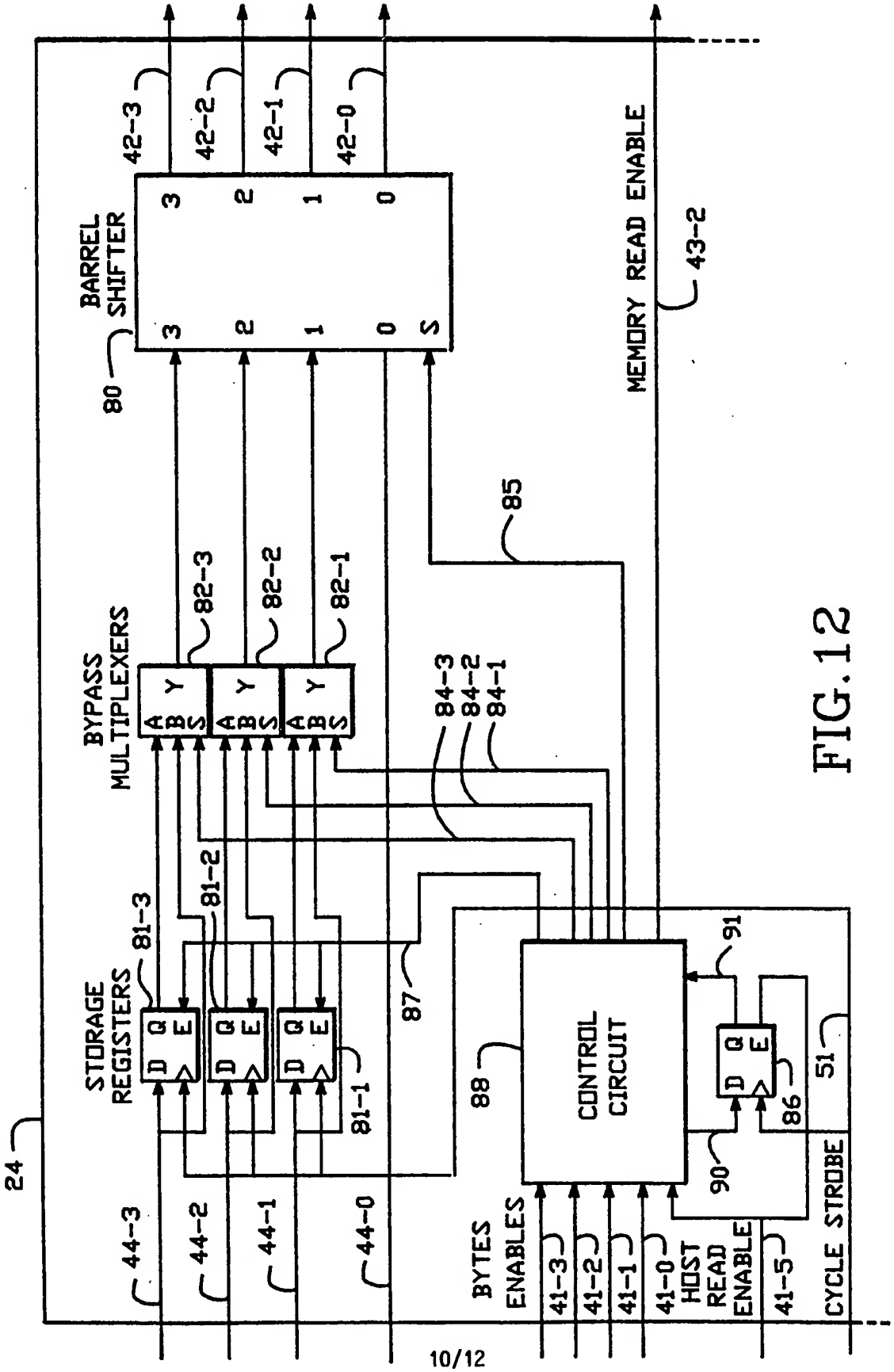

BARREL SHIFTER
80
42-3
42-2
42-1
42-0
3
2
1
0
S
MEMORY READ ENABLE
43-2
85
BYPASS MULTIPLEXERS
82-3
82-2
82-1
A
B
S
Y
84-3
84-2
84-1
STORAGE REGISTERS
81-3
81-2
81-1
D
Q
E
87
CONTROL CIRCUIT
88
91
90
86
51
24
44-3
44-2
44-1
44-0
BYTES ENABLES
41-3
41-2
41-1
41-0
HOST READ ENABLE
41-5
CYCLE STROBE

FIG. 12

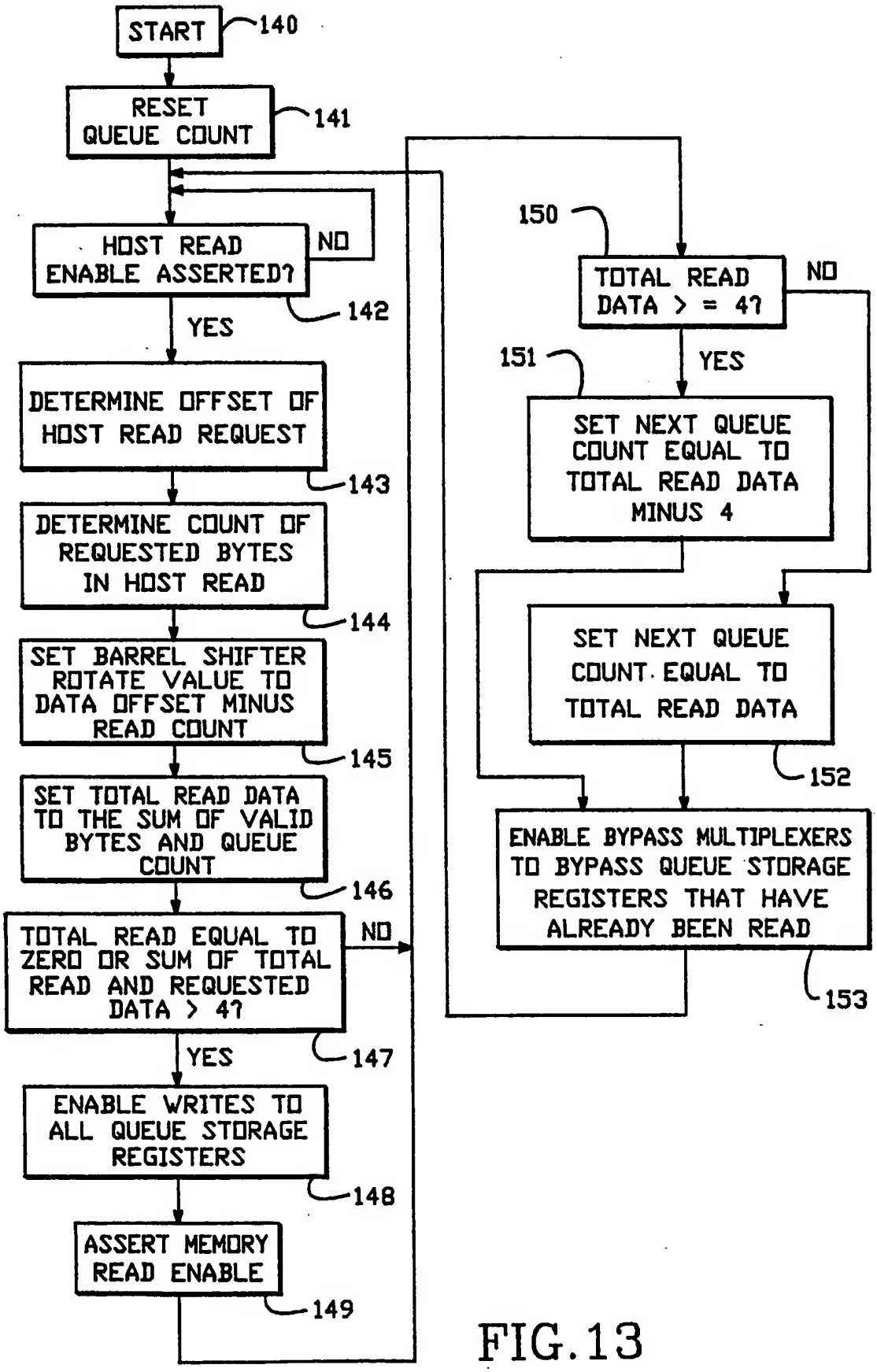

START
140
RESET QUEUE COUNT
141
HOST READ ENABLE ASSERTED?
NO
142
YES
DETERMINE OFFSET OF HOST READ REQUEST
143
DETERMINE COUNT OF REQUESTED BYTES IN HOST READ
144
SET BARREL SHIFTER ROTATE VALUE TO DATA OFFSET MINUS READ COUNT
145
SET TOTAL READ DATA TO THE SUM OF VALID BYTES AND QUEUE COUNT
146
TOTAL READ EQUAL TO ZERO OR SUM OF TOTAL READ AND REQUESTED DATA > 4?
NO
147
YES
ENABLE WRITES TO ALL QUEUE STORAGE REGISTERS
148
ASSERT MEMORY READ ENABLE
149
150
TOTAL READ DATA > = 4?
NO
151
YES
SET NEXT QUEUE COUNT EQUAL TO TOTAL READ DATA MINUS 4
SET NEXT QUEUE COUNT EQUAL TO TOTAL READ DATA
152
ENABLE BYPASS MULTIPLEXERS TO BYPASS QUEUE STORAGE REGISTERS THAT HAVE ALREADY BEEN READ
153

FIG.13

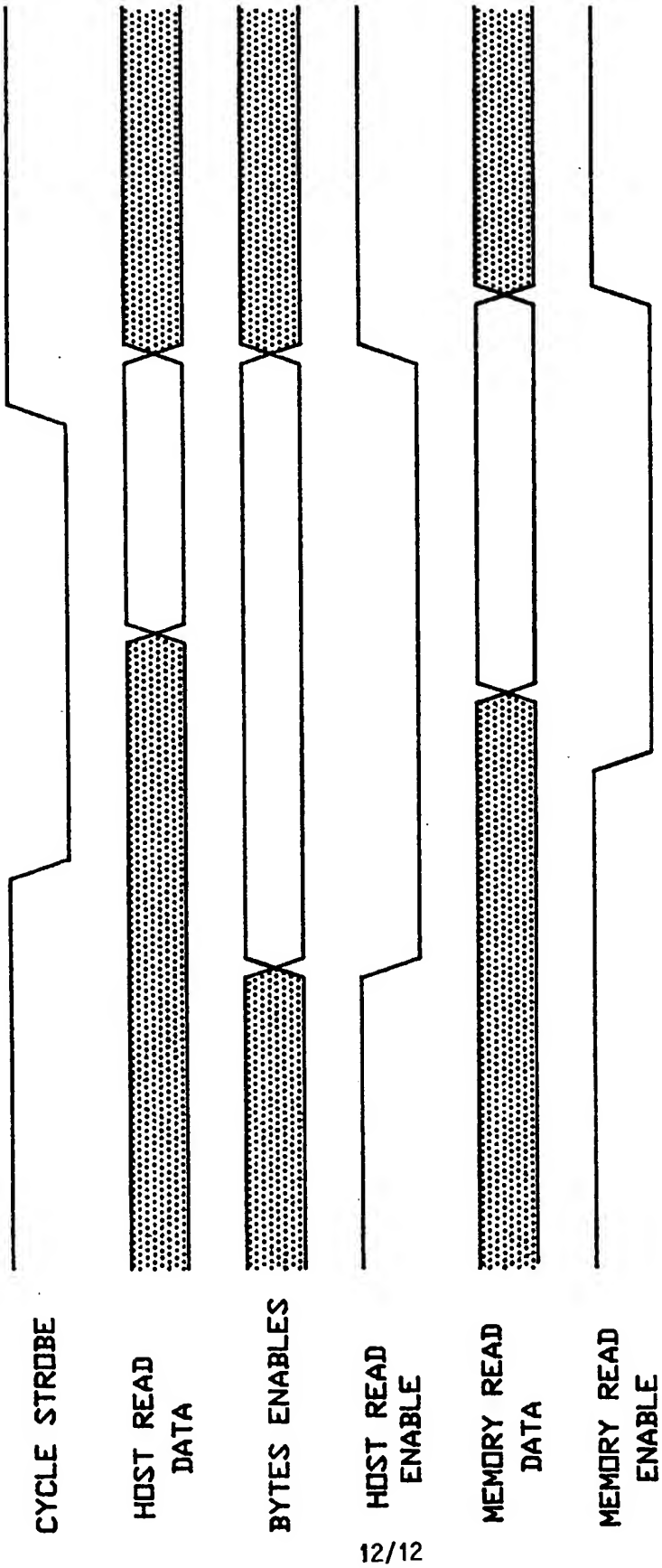

CYCLE STROBE
HOST READ DATA
BYTES ENABLES
HOST READ ENABLE
MEMORY READ DATA
MEMORY READ ENABLE

FIG. 14